ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computer and System Sciences 70 (2005) 176–200

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

A probabilistic model of computing with words☆

Daowen Qiu[a, b, *], Huaiqing Wang[c]

[a] *Department of Computer Science, Zhongshan University, Guangzhou 510275, P. R. China*
[b] *State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China*
[c] *Department of Information Systems, City University of Hong Kong, Hong Kong*

Received 1 February 2003; received in revised form 26 August 2004

Abstract

Computing in the traditional sense involves inputs with strings of numbers and symbols rather than *words*, where words mean probability distributions over input alphabet, and are different from the words in classical formal languages and automata theory. In this paper our goal is to deal with probabilistic finite automata (PFAs), probabilistic Turing machines (PTMs), and probabilistic context-free grammars (PCFGs) by inputting strings of words (probability distributions). Specifically, (i) we verify that PFAs computing strings of words can be implemented by means of calculating strings of symbols (Theorem 1); (ii) we elaborate on PTMs with input strings of words, and particularly demonstrate by describing Example 2 that PTMs computing strings of words may not be directly performed through only computing strings of symbols, i.e., Theorem 1 may not hold for PTMs; (iii) we study PCFGs and thus PRGs with input strings of words, and prove that Theorem 1 does hold for PCFRs and PRGs (Theorem 2); a characterization of PRGs in terms of PFAs, and the equivalence between PCFGs and their Chomsky and Greibach normal forms, in the sense that the inputs are strings of words, are also presented. Finally, the main results obtained are summarized, and a number of related issues for further study are raised.

Keywords: Probabilistic finite automata; Probabilistic Turing machines; Probabilistic context-free grammars; Strings of words; Computing with words; Quantum automata; Quantum models of computation

☆ This research is supported by the National Natural Science Foundation (No. 90303024), the Natural Science Foundation of Guangdong Province (No. 020146, 031541) of China, and the UGC CERG research grants (No. 9040451 and 9040708).
* Corresponding author.
E-mail address: issqdw@zsu.edu.cn (D. Qiu).

0022-0000/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2004.08.006

1. Introduction

Computing in the traditional sense involves inputs with strings of numbers and symbols rather than *words*, and even in those nontraditional models of computation such as molecular, stochastic, analog, and quantum computing [1,20,18,31,8,7,14,35], the inputs still are strings of numbers and symbols instead of *words*. In this paper, words mean probability distributions over input alphabet, and are different from the words in classical formal languages and automata theory [17] that indeed represent strings of input symbols. Motivated by the idea of computing with words (CW, for short) proposed and advocated recently by Zadeh [37–39], in this paper our goal is to deal with probabilistic finite automata (PFAs), probabilistic Turing machines (PTMs), and probabilistic context-free grammars (PCFGs) by inputting strings of words (probability distributions).

As a methodology, computing with words likely provides a foundation for a computational theory of perceptions on how machines make perception-based rational decisions in an environment of imprecision, uncertainty, and partial truth. Since CW was put forward, this issue has received extensive attention in the research community, and indeed there have been considerable discussions and much literature on linguistic variables and their applications to approximate reasoning, but most of these are irrelevant to the formal theory of computing. Recently, Ying [36] took a different viewpoint, in contrast with the conventional idea of CW. In Ying's view, the basic starting point is to deal with fuzzy finite automata and fuzzy pushdown automata by extending their inputs to include strings of fuzzy subsets of input alphabet. Those fuzzy subsets of input alphabet are indeed possibility distributions over input alphabet, and therefore can be viewed as words. Then, in [34], we continued to develop and deepen this formal aspect of CW by elaborating on fuzzy Turing machine with input strings of words and the equivalences between fuzzy computational models, in the sense that the inputs are of strings of words instead of symbols. Hence, to a certain extent, a formal aspect of CW has been given preliminary consideration [36,34].

In [36,34] words, as some fuzzy constraint variables, are treated as possibility distributions. However, words are sometimes processed necessarily as probabilistic constraint variables, because describing the real-world also needs probabilistic methods, and, in essence, probability theory has been being employed with remarkable success in those fields in which the systems are mechanistic; for example, statistical mechanics, quantum mechanics, communication systems and evolutionary programming, and related fields. Naturally, probabilistic models of computation, such as probabilistic automata, probabilistic Turing machines, and probabilistic grammars, are suitable formal computational models for words with probability.

With the motivation stated above, the present paper is to study PFAs, PTMs, and PCFGs in the sense that the inputs are strings of words. PFAs [28,23,29], as the simplest models of probabilistic computation, are a generalization of deterministic finite automata, and have significant applications to other disciplines such as reliability, learning theory and pattern recognition, and stochastic networks [28,12,31]. Though (1-way) probabilistic finite automata recognizes only regular languages under the acceptance way of bounded-error probability [28, 23, p. 160], two-way probabilistic automata can recognize non-regular languages [11]. As well, PFAs can be viewed as an especial case of quantum finite automata [19,2,3,5,15,24,25]. PTMs [9,30,10,13,22], a generalized type of deterministic Turing machines, serve as a formal model for randomized algorithms [20] and the study of the potentials and limitations of computing, and lay an important foundation for the modern Church–Turing thesis and computational complexity [22]. As well, the underlying models of quantum computation–quantum Turing machines [8,7] can be thought of

as generalizations of PTMs to a certain extent. PCFGs are widely studied [6,12,32,33] and significantly applied to the analysis of programming languages, automatic parses, error correctors [4,33], and frequently used in syntactic pattern recognition [12] and computational linguistic, as well as speech recognition and understanding [21]. Considering these key roles played by the three models of computation, from theoretical study to practical applications, we hence choose them as the underlying formal models for inputting strings of words.

The major technical contributions of this paper are twofold. On the one hand, we study PFAs, PTMs, and PCFGs with the inputs to be strings of words that are probability distributions over input alphabets. In particular, we discover that PTMs computing strings of words may not be derived from computing strings of symbols, while we demonstrate such a property does hold true for PFAs and PCFGs computing strings of words. Also, we find that the computational complexity of these computational models for computing strings of words likely increases exponentially in comparison with computing strings of symbols. On the other hand, we may provide a formal approach to CW in the sense of probability, as fuzzy finite automata and fuzzy Turing machines with input strings of words investigated in [36,34]. As well, we briefly compare our results obtained in this paper with those in [36,34]. It is worth mentioning that these established models may have applications in other fields, for example, syntactic pattern recognition and programming languages.

The remainder of the paper is organized as follows. In Section 2, we deal with PFAs with input strings of words. Specifically, we can derive the probability for PFAs computing strings of words by means of calculating these probabilities of PFAs recognizing strings of symbols (Theorem 1), and thus shows that PFAs for inputting strings of words can be transferred to PFAs computing the conventional strings of symbols, at the expense of more steps of computation, which may be called *computation tractability*.

In Section 3, we elaborate on PTMs with input strings of words. We first define a general type of PTMs, in which every move is allowed to have multiple choices, and different computations for a fixed input may have distinct numbers of steps, while in standard PTMs [13,22,20], each move has only two choices and all computations for a given input have the same length. Then we discuss the computation tractability of PTMs with input strings of words, corresponding to Theorem 1, and we find that PTMs computing strings of words may not be processed by means of calculating all strings of symbols (Example 2). This is a negative result, revealing a characteristic of PTMs for inputting strings of words different from others with input strings of words such as PFAs and PCFGs.

In Section 4, we consider PCFGs and PRGs with input strings of words. The computation tractability theorem (Theorem 2) for PCFGs and thus PRGs with strings of words is proved. As well, we demonstrate a characterization of PRGs in terms of PFAs, and the equivalence between PCFGs and their Chomsky and Greibach normal forms, in the sense that the inputs are strings of words.

Finally, in Section 5 we summarize the main results obtained, and briefly compare them with those in [36,34], as well as present a number of related issues for further study.

Notations we use in this paper will be explained when they first arise.

2. Probabilistic finite automata with input strings of words

In this section, we consider the issue of (1-way) PFAs with input strings of words, as a start of investigating probabilistic models for inputting strings of words.

Definition 1. A *probabilistic finite automaton* (PFA) $\mathcal{A}$ is 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is the finite input alphabet, q_0 is the initial state viewed as a probability distribution over Q (e.g., $q_0 = \sum_{q_i \in Q} a_i q_i$ with $\sum a_i = 1$ and $a_i \in (0, 1]$), $F \subseteq Q$ is the accepting states, and δ is the probabilistic transition function, that is, $\delta : Q \times \Sigma \times Q \to [0, 1]$ satisfying

$$\sum_{q \in Q} \delta(p, \sigma, q) = 1 \tag{1}$$

for any $p \in Q$ and $\sigma \in \Sigma$.

In particular, if the range of δ is $\{0, 1\}$ and $q_0 \in Q$, then the above defined PFA is a *deterministic finite automaton* (DFA).

The *language* recognized by above PFA $\mathcal{A}$ is defined to be a function $L_{\mathcal{A}} : \Sigma^* \to [0, 1]$ as follows: for any $x = \sigma_1\sigma_2 \ldots \sigma_k \in \Sigma^*$, where Σ^* stands for the set of all strings over Σ, containing empty string ε,

$$L_{\mathcal{A}}(x) = \sum \{\delta(q_0, \sigma_1, q_1) \cdot \delta(q_1, \sigma_2, q_2) \cdot \ldots \cdot \delta(q_{k-1}, \sigma_k, q) : q \in F, q_i \in Q, i = 1, 2, \ldots, k-1\}, \tag{2}$$

where $\delta(q_0, \sigma_1, q_1) = \sum a_i \delta(q_i, \sigma_1, q_1)$ if $q_0 = \sum a_i q_i$.

A word over alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, according to Zadeh's opinion [38], can be defined as a probability distribution W over Σ, and it is denoted by

$$W = a_1\backslash\sigma_1 + a_2\backslash\sigma_2 + \cdots + a_m\backslash\sigma_m,$$

where $W(\sigma_i) = a_i$, $i = 1, 2, \ldots, m$. The set of words over Σ is denoted by $\mathcal{D}(\Sigma)$. Notably, according to Zadeh [38], *possibility distributions* are represented as the form

$$W = a_1/\sigma_1 + a_2/\sigma_2 + \cdots + a_m/\sigma_m$$

which is different from the above representation for *probability distributions*. As we indicated above, in this paper, words are viewed as *probability distributions*.

We now consider the inputs to be strings of words instead of symbols, that is, strings of probability distributions over input alphabet. First we extent δ from $Q \times \Sigma \times Q$ to $Q \times \mathcal{D}(\Sigma) \times Q$. For any words $W \in \mathcal{D}(\Sigma)$, $p, q \in Q$,

$$\delta(p, W, q) = \sum_{\sigma \in \Sigma} \delta(p, \sigma, q) W(\sigma).$$

Furthermore, for any string of words $W_1 W_2 \cdots W_k \in \mathcal{D}(\Sigma)^*$,

$$L_{\mathcal{A}}(W_1 W_2 \ldots W_k) = \sum \left\{ \delta(q_0, W_1, q_1) \cdot \delta(q_1, W_2, q_2) \cdot \ldots \cdot \delta(q_{k-1}, W_k, q) : q \in F, q_i \in Q, i = 1, 2, \ldots, k-1 \right\}.$$

To simplify the computation of $L_{\mathcal{A}}(W_1W_2\cdots W_k)$, we hope to derive $L_{\mathcal{A}}(W_1W_2\cdots W_k)$ by computing $L_{\mathcal{A}}(\sigma_1\sigma_2\ldots\sigma_k)$. Indeed, we have the following result that may be called *computation tractability*.

Theorem 1. *For any* PFA $\mathcal{A}=(Q,\Sigma,\delta,q_0,F)$*, and any string of words* $W=W_1W_2\cdots W_k$*, we have*

$$L_{\mathcal{A}}(W)=\sum_{\sigma_1,\sigma_2,\ldots,\sigma_k\in\Sigma} L_{\mathcal{A}}(\sigma_1\sigma_2\cdots\sigma_k)\times\left(\prod_{i=1}^{k}W_i\right)(\sigma_1\sigma_2\cdots\sigma_k),$$

where, also, in what follows, $(\prod_{i=1}^{k}W_i)(\sigma_1\sigma_2\cdots\sigma_k)\overset{def}{=}W_1(\sigma_1)\cdot W_2(\sigma_2)\cdot\ldots\cdot W_k(\sigma_k)$.

Proof. For any strings of words $W=W_1W_2\ldots W_k\in\mathcal{D}(\Sigma)^*$, we have

$$\begin{aligned}
&L_{\mathcal{A}}(W_1W_2\ldots W_k)\\
&\quad=\sum\{\delta(q_0,W_1,q_1)\cdot\delta(q_1,W_2,q_2)\cdot\ldots\cdot\delta(q_{k-1},W_k,q):\\
&\qquad q\in F,q_i\in Q,i=1,2,\ldots,k-1\}\\
&\quad=\sum\left\{\left(\sum_{\sigma_1\in\Sigma}\delta(q_0,\sigma_1,q_1)\cdot W_1(\sigma_1)\right)\left(\sum_{\sigma_2\in\Sigma}\delta(q_1,\sigma_2,q_2)\cdot W_2(\sigma_2)\right)\cdot\ldots\cdot\right.\\
&\qquad\left.\times\left(\sum_{\sigma_k\in\Sigma}\delta(q_{k-1},\sigma_k,q)\cdot W_k(\sigma_k)\right):q\in F,q_i\in Q,i=1,2,\ldots,k-1\right\}\\
&\quad=\sum_{\sigma_1,\sigma_2,\ldots,\sigma_k}W_1(\sigma_1)\cdot W_2(\sigma_2)\cdot\ldots\cdot W_k(\sigma_k)\\
&\qquad\times\sum_{q\in F,q_i\in Q}\delta(q_0,\sigma_1,q_1)\cdot\delta(q_1,\sigma_2,q_2)\cdot\ldots\cdot\delta(q_{k-1},\sigma_k,q)\\
&\quad=\sum_{\sigma_1,\sigma_2,\ldots,\sigma_k}L_{\mathcal{A}}(\sigma_1\sigma_2\ldots\sigma_k)\cdot\left(\prod_{i=1}^{k}W_i\right)(\sigma_1\sigma_2\cdots\sigma_k).\qquad\square
\end{aligned}$$

Remark 1. Let us observe the time complexity for a PFA computing strings of words in comparison to computing strings of symbols. For any PFA $\mathcal{A}=(Q,\Sigma,\delta,q_0,F)$ and any string $x\in\Sigma^*$ with length $|x|=n$, then from Eq. (1) the steps of computing $L_{\mathcal{A}}(x)$ (time complexity) is at most $T_{\mathcal{A}}(n)=(n+1)|Q|^{n+1}$ relying on n but independent of x. In contrast, if the input is a string of words, say $W=W_1W_2\cdots W_n\in\mathcal{D}(\Sigma)^*$, then from Theorem 1 it follows readily that the steps of computing $L_{\mathcal{A}}(W)$ is at most and possibly $(n+1)|\Sigma|^nT_{\mathcal{A}}(n)$, which shows that the time complexity of computing strings of words may increase exponentially when it is compared with computing strings of symbols, in case $|\Sigma|>1$.

We now describe an example to illustrate the application of Theorem 1.

Example 1. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a PFA, where $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $q_0 = \frac{1}{2}q_1 + \frac{1}{2}q_3$, $F = \{q_3\}$ and δ is defined as follows:

$$\begin{aligned}
&\delta(q_1, a, q_1) = \tfrac{1}{2},\ \delta(q_1, a, q_3) = \tfrac{1}{2},\\
&\delta(q_2, a, q_2) = 1,\ \delta(q_3, a, q_1) = \tfrac{1}{3},\\
&\delta(q_3, a, q_3) = \tfrac{2}{3},\ \delta(q_1, b, q_2) = 1,\\
&\delta(q_2, b, q_1) = \tfrac{1}{4},\ \delta(q_2, b, q_3) = \tfrac{3}{4},\\
&\delta(q_3, b, q_3) = 1;
\end{aligned}$$

otherwise, $\delta(p, x, q) = 0$. Then

$$L_{\mathcal{A}}(aa) = \sum_{p\in Q} \delta(q_0, a, p)\delta(p, a, q_3) = \frac{43}{72},$$

$$L_{\mathcal{A}}(ab) = \sum_{p\in Q} \delta(q_0, a, p)\delta(p, b, q_3) = \frac{7}{12},$$

$$L_{\mathcal{A}}(ba) = \sum_{p\in Q} \delta(q_0, b, p)\delta(p, a, q_3) = \frac{1}{3},$$

$$L_{\mathcal{A}}(bb) = \sum_{p\in Q} \delta(q_0, b, p)\delta(p, b, q_3) = \frac{3}{4}.$$

Let the probability distributions W_1, W_2 over $\{a, b\}$ be defined as:

$$W_1 = 0.4\backslash a + 0.6\backslash b, \quad W_2 = 0.7\backslash a + 0.3\backslash b.$$

Then, by Theorem 1 we obtain that

$$L_{\mathcal{A}}(W_1W_2) = \sum_{\sigma_1,\sigma_2} W_1(\sigma_1)W_2(\sigma_2)L_{\mathcal{A}}(\sigma_1\sigma_2) = \frac{461}{900};$$

$$L_{\mathcal{A}}(W_2W_1) = \sum_{\sigma_1,\sigma_2} W_2(\sigma_1)W_1(\sigma_2)L_{\mathcal{A}}(\sigma_1\sigma_2) = \frac{131}{225}.$$

Remark 2. Due to the potential of applications of probabilistic pushdown automata (PPAs) [12], a natural issue is to deal with PPAs with input strings of words and establish the relation between PPAs and PCFGs in the sense that the inputs are strings of words. However, with regard to our purposes of this paper principally centering on PFAs, PTMs and PCFGs with input strings of words, the details will be processed in other place. To conclude this section, we would like to propose a problem: whether Theorem 1 holds true for two-way PFAs [11]?

3. Probabilistic turing machines with input strings of words

PTMs were first considered by De Leeuw et al. [9], Santos [30], and Ellis [10]. Then Gill, Simon, and others (see [22]) studied the computational complexity of PTMs. In this section, we focus on PTMs with input strings of words. More specifically, we define a general type of PTMs, and consider the inputs into PTMs to include strings of words. In particular, we obtain a negative result corresponding to Theorem 1; that is to say, Theorem 1 (Eq. (3)) may not hold true if the underlying models of computation are PTMs rather than PFAs.

In spite of the different variants of PTMs defined in the literature, they intrinsically coincide in spirit. A simple version of PTMs [22,13] that are convenient for the study of polynomial time randomized complexity classes is with *coin-flipping* states, more specifically, states in which (1) every step of the computation can only have two outcomes, that is, every configuration has at most two next configurations with respective probabilities p and $1-p$; (2) all computations on the same input end with the same number of steps; (3) and every computation ends with either *reject* or *accept*. Thus, for a probabilistic Turing machine to be *coin-flipping* it is defined usually as the range of its transition function being $\{0, \frac{1}{2}, 1\}$. In reality, we call such a type of models to be *standard* PTMs, which is frequently considered, for example, in the study of randomized algorithms and computational complexity [20,22,13].

In the present paper, we define a quite general form of PTMs, which generalize deterministic TMs (DTMs) and are formally analogous to nondeterministic Turing machines [17].

Formally, a DTM (two-way and one-tape) is a system denoted by $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where Q is the finite set of states; Σ is the finite set of input symbols; Γ is the finite set of tape symbols, containing Σ and an identified blank symbol $\# \notin \Sigma$ as well as other allowable tape symbols; q_0, q_a, q_r in Q are the *starting*, *accepting*, and *rejecting* states, respectively; and δ that describes the dynamics (the computation) of the machine is a mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$ where L and R decide the directions for the *read–write tape-head* to move. However, it is possible that $\delta(q, x)$ may be undefined for some $(q, x) \in Q \times \Gamma$, which results in the machine halting with neither accepting nor rejecting configurations.

A *configuration* or *instantaneous description* of a TM M is described by a string as $\alpha_1 q \alpha_2$ for $q \in Q$ and $\alpha_1\alpha_2 \in \Gamma^*$ where Γ^* denotes the set of all these finite strings over Γ including empty string ε, the leftmost and the rightmost symbols in configuration $\alpha_1 q \alpha_2$ are not the blank #, and the *read–write tape-head* is scanning the leftmost symbol of α_2 or the blank # in case $\alpha_2 = \varepsilon$. The moves of the *read–write tape-head* depend on the transition function δ. In fact, δ may be equivalently described as a *program* that is defined to be made up of those *program lines* or called *instructions* of the form:

$$\langle q, x, q', x', d\rangle,$$

where $(q, x) \in Q \times \Gamma$ and $\delta(q, x) = (q', x', d)$. In case $\delta(q, x)$ is undefined, then a *halting instruction* results. The way the program works is that on each machine cycle, the machine looks through the list of *program lines* in an orderly way, searching for a line $\langle q, x, ., ., .\rangle$, such that the current state is q and, the symbol being scanned by the *read–write tape-head* is x. If a *program line* for instance $\langle q, x, q', x', L\rangle$ is found, then it is executed, that is, the current control state q is changed to q', the read symbol is rewritten by x' and, the *read–write tape-head* moves left. As we know, a *program* contains a definite *starting instruction*, but it is not intrinsic for our discussion, since we can formally add such an instruction as

$$\langle q_s, \triangleright, q_0, \triangleright, R\rangle,$$

where q_s denotes a starting state different from those in Q; $\triangleright$, a special tape symbol, lies in the left of the input strings. If so, the initial configuration for inputting w is thus as $q_0 \triangleright w$.

More clearly, we describe a move of the above TM M from a configuration c_1 to another one, say c_2, which is denoted by $c_1 \hookrightarrow_M c_2$, by dividing it into three cases:

Case 1: If $c_1 = qx_1x_2\cdots x_k$, $x_i \in \Gamma$, $i = 1, 2, \ldots, k$, then

$$c_2 = \begin{cases} ypx_2x_3\cdots x_k, & \text{if } \delta(q, x_1) = (p, y, R); \\ p\#yx_2\cdots x_k, & \text{if } \delta(q, x_1) = (p, y, L). \end{cases}$$

Case 2: If $c_1 = x_1x_2\cdots x_{i-1}qx_i\cdots x_k$, $2 \leqslant i \leqslant n$, $x_i \in \Gamma$, $i = 1, 2, \ldots, k$, then

$$c_2 = \begin{cases} x_1x_2\cdots x_{i-1}ypx_{i+1}\cdots x_k, & \text{if } \delta(q, x_i) = (p, y, R); \\ x_1\cdots px_{i-1}yx_{i+1}\cdots x_k, & \text{if } \delta(q, x_i) = (p, y, L). \end{cases}$$

Case 3: If $c_1 = x_1x_2\cdots x_kq$, $x_i \in \Gamma$, $i = 1, 2, \ldots, k$, then

$$c_2 = \begin{cases} x_1x_2\cdots x_kyp, & \text{if } \delta(q, \#) = (p, y, R); \\ x_1x_2\cdots x_{k-1}px_ky, & \text{if } \delta(q, \#) = (p, y, L). \end{cases}$$

To define the language recognized by the above machine M, we denote by $C_a(M)$, $C_r(M)$, and $C_h(M)$ the sets of accepting configurations, rejecting configurations, and halting configurations, respectively, which are defined as:

$$C_a(M) = \{\alpha q_a \beta : \alpha\beta \in \Gamma^*\}, \tag{3}$$

$$C_r(M) = \{\alpha q_r \beta : \alpha\beta \in \Gamma^*\}, \tag{4}$$

$$C_h(M) = \{\alpha q x \beta : q \in Q - \{q_a, q_r\}, \alpha\beta \in \Gamma^*, \delta(q, x) \text{ is undefined}\}. \tag{5}$$

Denote $\hookrightarrow_M^*$ to be the reflexive and transitive closure of $\hookrightarrow_M$. Then the *languages* accepted and rejected by Turing machine M, are respectively the sets

$$L_a(M) = \{w : w \in \Sigma^*, q_0w \hookrightarrow_M^* c_a, c_a \in C_a(M)\},$$

$$L_r(M) = \{w : w \in \Sigma^*, q_0w \hookrightarrow_M^* c_r, c_r \in C_r(M)\}.$$

In the model described above, if the program is *probabilistic*, and the changed states, the rewritten symbols and the moved directions have multiple choices with respective probabilities, then we are led to defining naturally a PTM, which is formally described as follows.

Definition 2. A PTM is a system $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where $q_0, q_a, q_r \in Q$ are the starting, accepting, and rejecting states, respectively, and

$$\delta : Q \times \Gamma \times Q \times \Gamma \times \{L, R\} \to [0, 1]$$

is the transition function satisfying that for all $(q, x) \in Q \times \Gamma$, it is either

$$\sum_{(q', x', d) \in Q \times \Sigma \times \{L, R\}} \delta(q, x, q', x', d) = 1,$$

or $\delta(q, x, \cdot, \cdot, \cdot) = 0$ that represents $\delta(q, x, q', x', d) = 0$ for all $(q', x', d) \in Q \times \Gamma \times \{L, R\}$.

Intuitively, $\delta(q, x, q', x', d)$ represents the probability that the current control state q and the tape symbol x being scanned are capable of turning to state q' and rewriting tape symbol x', together with moving left (when $d = L$) or right (when $d = R$).

The definitions of configurations, halting configurations, accepting configurations, and rejecting configurations of PTMs are similar to those of deterministic TMs present as in Eqs. (3)–(5), defined as:

$\mathcal{C}_{\rm a}(\mathcal{M}) = \{\alpha q_{\rm a}\beta : \alpha, \beta \in \Gamma^*\}$,

$\mathcal{C}_{\rm r}(\mathcal{M}) = \{\alpha q_{\rm r}\beta : \alpha, \beta \in \Gamma^*\}$, and

$\mathcal{C}_{\rm h}(\mathcal{M}) = \{\alpha q x \beta : (q, x) \in H, \alpha, \beta \in \Gamma^*\}$,

where $H = \{(q, x) : q \in Q - \{q_{\rm r}, q_{\rm a}\}, x \in \Gamma, \delta(q, x, \cdot, \cdot, \cdot) = 0\}$ stands for the set of all pairs (q, x) such that in current state $q \in Q - \{q_{\rm r}, q_{\rm a}\}$ and tape symbol x, the machine will halt with neither accepting nor rejecting state. Actually, accepting and rejecting configurations represent also a kind of "halting configurations", which means that when the machine moving to accepting or rejecting configuration, a process of computation will halt. But here they are phrased with only distinct names for the convenience of further statement.

The actions of the PTMs defined above are able to be described equivalently with a *probabilistic program*, and such a probabilistic program consists of some *probabilistic program lines* or called *probabilistic instructions* of the form

$$L_{ij}(p_{ij}) : \langle q_i, x, q', x', d\rangle \tag{6}$$

for $1 \leqslant j \leqslant k_i$, where $L_{ij}(p_{ij})$ denotes just a mark, the subscript i coincides with that of state q_i, and k_i is the number of all program lines relating to their current state q_i, $p_{ij} = \delta(q_i, x, q', x', d)$ stands for the probability explained above.

On PTM $\mathcal{M}$, a move from configuration c_1 to another c_2 by performing a probabilistic program line $L_{ij}(p_{ij})$ is thus denoted by

$$c_1 \overset{L_{ij}(p_{ij})}{\hookrightarrow_{\mathcal{M}}} c_2$$

and the corresponding probability $Pr(c_1 \overset{L_{ij}(p_{ij})}{\hookrightarrow_{\mathcal{M}}} c_2)$ is equal to p_{ij}. In general, a configuration c_1 moving to another c_2 is simply denoted by $c_1 \hookrightarrow_{\mathcal{M}} c_2$, and its probability can be defined in terms of the transition function δ in the following way:

$$Pr(c_1 \hookrightarrow_{\mathcal{M}} c_2) = \begin{cases} \delta(q_i, x_1, q_t, y, R), & \text{if } c_1 = q_i x_1 x_2 \cdots x_k, c_2 = y q_t x_2 x_3 \cdots x_k, \\ \delta(q_i, x_1, q_t, y, L), & \text{if } c_1 = q_i x_1 x_2 \cdots x_k, c_2 = q_t \# y x_2 x_3 \cdots x_k, \\ \delta(q_i, x_i, q_t, y, R), & \text{if } c_1 = x_1 x_2 \cdots x_{i-1} q_i x_i \cdots x_k, 2 \leqslant i \leqslant n, \\ & \quad c_2 = x_1 x_2 \cdots x_{i-1} y q_t x_{i+1} \cdots x_k, \\ \delta(q_i, x_i, q_t, y, L), & \text{if } c_1 = x_1 x_2 \cdots x_{i-1} q_i x_i \cdots x_k, 2 \leqslant i \leqslant n, \\ & \quad c_2 = x_1 x_2 \cdots x_{i-2} q_t x_{i-1} y x_{i+1} \cdots x_k, \\ \delta(q_i, \#, q_t, y, R), & \text{if } c_1 = x_1 x_2 \cdots x_k q_i, c_2 = x_1 x_2 \cdots x_k y q_t, \\ \delta(q_i, \#, q_t, y, L), & \text{if } c_1 = x_1 x_2 \cdots x_k q_i, c_2 = x_1 x_2 \cdots x_{k-1} q_t x_k y, \\ 0, & \text{otherwise}, \end{cases}$$

in which each different configuration pair (c_1, c_2) is associated with a certain probabilistic instruction $L_{ij}(p_{ij})$, but i is the same since their current state is exactly q_i. Intuitively, for any starting configuration $c_0 = q_0 x$ where x is an input string, all possible moves of the machine will then form a configuration

tree in which c_0 is viewed as the root and the halting, accepting, and rejecting configurations as leaves. Therefore, each computation is a path from the root to a leaf. For the details, we will describe them in Example 2.

More formally, a *computation* for inputting string $w \in \Sigma^*$ of length k with time $t(k)$ and space $s(k)$ is defined as a path of transitions described by:

$$c_0 \overset{L_{i_1 j_1}(p_{i_1 j_1})}{\hookrightarrow_{\mathcal{M}}} c_1 \overset{L_{i_2 j_2}(p_{i_2 j_2})}{\hookrightarrow_{\mathcal{M}}} \cdots \overset{L_{i_m j_m}(p_{i_m j_m})}{\hookrightarrow_{\mathcal{M}}} c_m,$$

and the corresponding probability denoted by

$$Pr(c_0 \overset{L_{i_1 j_1}(p_{i_1 j_1})}{\hookrightarrow_{\mathcal{M}}} c_1 \overset{L_{i_2 j_2}(p_{i_2 j_2})}{\hookrightarrow_{\mathcal{M}}} \cdots \overset{L_{i_m j_m}(p_{i_m j_m})}{\hookrightarrow_{\mathcal{M}}} c_m)$$

is equal to the product of all probabilities for those related moves, i.e.,

$$\prod_{k=1}^{m} Pr(c_{k-1} \overset{L_{i_k j_k}(p_{i_k j_k})}{\hookrightarrow_{\mathcal{M}}} c_k) = \prod_{k=1}^{m} p_{i_k j_k},$$

where $m = t(k)$ and none of configurations c_i $(i = 0, 1, \ldots, m)$ is longer than $s(k)$, $c_0 = q_0 w$ and, c_m belongs to $\mathcal{C}_{\mathrm{a}}(\mathcal{M}) \cup \mathcal{C}_{\mathrm{r}}(\mathcal{M}) \cup \mathcal{C}_{\mathrm{h}}(\mathcal{M})$.

We denote by $\hookrightarrow^*_{\mathcal{M}}$ the reflexive and transitive closure of $\hookrightarrow_{\mathcal{M}}$; that is, $c_1 \hookrightarrow^*_{\mathcal{M}} c_2$ stands for all allowable paths from c_1 to c_2, and therefore its probability $Pr(c_1 \hookrightarrow^*_{\mathcal{M}} c_2)$ is the sum of those probabilities for all allowable different paths from c_1 to c_2. In particular, for natural number t, we use

$$c_1 \hookrightarrow^{*,t}_{\mathcal{M}} c_2$$

to represent all those allowable distinct paths from c_1 to c_2 in exact ts steps, i.e., containing t's edges, and the corresponding probability is denoted by $Pr(c_1 \hookrightarrow^{*,t}_{\mathcal{M}} c_2)$.

Then the *languages* accepted and rejected by the above PTM $\mathcal{M}$ are, respectively, defined as two functions $Pr^{\mathrm{acc}}_{\mathcal{M}}$ and $Pr^{\mathrm{rej}}_{\mathcal{M}}$ from Σ^* to $[0, 1]$: for any $w \in \Sigma^*$,

$$Pr^{\mathrm{acc}}_{\mathcal{M}}(w) = \sum_{t=1}^{\infty} \sum_{c_{\mathrm{a}} \in \mathcal{C}_{\mathrm{a}}(\mathcal{M})} Pr(q_0 w \hookrightarrow^{*,t}_{\mathcal{M}} c_{\mathrm{a}}), \tag{7}$$

$$Pr^{\mathrm{rej}}_{\mathcal{M}}(w) = \sum_{t=1}^{\infty} \sum_{c_{\mathrm{r}} \in \mathcal{C}_{\mathrm{r}}(\mathcal{M})} Pr(q_0 w \hookrightarrow^{*,t}_{\mathcal{M}} c_{\mathrm{r}}). \tag{8}$$

Remark 3. In a standard PTM $\mathcal{M}$ (coin-flipping moves) with input alphabet Σ and initial state q_0 as described above, since all computations for each given input $w \in \Sigma$ have the same number of computation steps, for example, denoted by $N(w)$, we have

$$Pr^{\mathrm{acc}}_{\mathcal{M}}(w) = \sum_{c_{\mathrm{a}} \in \mathcal{C}_{\mathrm{a}}(\mathcal{M})} Pr(q_0 w \hookrightarrow^{*,N(w)}_{\mathcal{M}} c_{\mathrm{a}}). \tag{9}$$

Our main purpose is to deal with the general PTMs $\mathcal{M}$ defined above with input strings of words, so, we need consider the devices established above by inputting strings of *words* instead of symbols. Recall that $\mathcal{D}(\Sigma)$ denotes the set of words (probability distributions over Σ). Letting $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}})$ be

a PTM, we first generalize transition function δ to $\delta : Q \times (\mathcal{D}(\Sigma) \cup \Gamma) \times Q \times \Gamma \times \{L, R\} \to [0, 1]$ in a natural way: for any $W \in \mathcal{D}(\Sigma)$, and any $q_i, q_k \in Q$, $y \in \Gamma$, $d \in \{L, R\}$,

$$\delta(q_i, W, q_k, y, d) = \sum_{\sigma \in \Sigma} \delta(q_i, \sigma, q_k, y, d) W(\sigma) \tag{10}$$

and the corresponding probabilistic instruction for inputting word W is denoted by

$$L_{ij}(W, p_{ij}) : \langle q_i, W, q_j, y, d \rangle \tag{11}$$

where $L_{ij}(W, p_{ij})$ is only a mark implying certain connection to q_i and word W, $p_{ij} = \delta(q_i, W, q_j, y, d)$, $1 \leqslant j \leqslant k_i(W)$ and $k_i(W)$ is the number of all different pairs (q_m, y, d) satisfying $\delta(q_i, W, q_m, y, d) \neq 0$ for given pair (q_i, W). In case $\delta(q_i, W, q_m, y, d) = 0$ for any pair (q_m, y, d), then it yields a halting instruction.

Now for any $\alpha, \beta \in (\mathcal{D}(\Sigma) \cup \Gamma)^*$ and $q \in Q$, the form $\alpha q \beta$ is called a *configuration with words*; furthermore, the sets of accepting, rejecting, and halting *configurations with words* are correspondingly defined respectively as:

$\mathcal{C}_{\mathrm{a}}(\mathcal{D}(\Sigma), \mathcal{M}) = \{\alpha q_{\mathrm{a}} \beta : \alpha, \beta \in (\mathcal{D}(\Sigma) \cup \Gamma)^*\}$,
$\mathcal{C}_{\mathrm{r}}(\mathcal{D}(\Sigma), \mathcal{M}) = \{\alpha q_{\mathrm{r}} \beta : \alpha, \beta \in (\mathcal{D}(\Sigma) \cup \Gamma)^*\}$, and
$\mathcal{C}_{\mathrm{h}}(\mathcal{D}(\Sigma), \mathcal{M}) = \{\alpha q X \beta : (q, X) \in \mathcal{H}, \alpha, \beta \in (\mathcal{D}(\Sigma) \cup \Gamma)^*\}$,

where $\mathcal{H} = \{(q, X) : q \in Q, X \in \mathcal{D}(\Sigma) \cup \Gamma, \delta(q, X, \cdot, \cdot, \cdot) = 0\}$, and, $\delta(q, X, \cdot, \cdot, \cdot) = 0$, similar to that indicated above, represents $\delta(q, X, q', y, d) = 0$ for any pair $(q', y, d) \in Q \times \Gamma \times \{L, R\}$.

A configuration with words c_1 moving to another, say c_2 in one step via performing a probabilistic instruction: $L_{ij}(W, p_{ij}) : \langle q_i, W, q_k, y, d \rangle$, is denoted as

$$c_1 \overset{L_{ij}(W, p_{ij})}{\hookrightarrow_{\mathcal{M}}} c_2$$

and the corresponding probability is as

$$p_{ij} = \delta(q_i, W, q_j, y, d).$$

Generally, a configuration with words c_1 moving to another c_2 is still denoted by $c_1 \hookrightarrow_{\mathcal{M}} c_2$ as that used above for moving conventional configurations, and its probability is as follows: for any $W_1, W_2 \in \mathcal{D}(\Sigma) \cup \Gamma$, $\alpha, \beta \in (\mathcal{D}(\Sigma) \cup \Gamma)^*$, and $y \in \Gamma$,

$$Pr(c_1 \hookrightarrow_{\mathcal{M}} c_2) = \begin{cases} \delta(q_i, W_1, q_t, y, R), & c_1 = q_i W_1 \alpha, c_2 = y q_t \alpha, \\ \delta(q_i, W_1, q_t, y, L), & c_1 = q_i W_1 \alpha, c_2 = q_t \# y \alpha, \\ \delta(q_i, W_2, q_t, y, R), & c_1 = \alpha W_1 q_i W_2 \beta, c_2 = \alpha W_1 y q_t \beta, \\ \delta(q_i, W_2, q_t, y, L), & c_1 = \alpha W_1 q_i W_2 \beta, c_2 = \alpha q_t W_1 y \beta, \\ \delta(q_i, \#, q_t, y, R), & c_1 = \alpha W_1 q_i, c_2 = \alpha W_1 y q_t, \\ \delta(q_i, \#, q_t, y, L), & c_1 = \alpha W_1 q_i, c_2 = \alpha q_t W_1 y, \\ 0, & \text{otherwise}, \end{cases}$$

notably, in which W_1, W_2, besides being some words, can also be replaced by any tape symbols belonging to Γ.

Given a string of words $W = W_1 W_2 \cdots W_m$ where $W_i \in \mathcal{D}(\Sigma)$, $i = 1, 2, \ldots, m$, a *computation* in t steps for inputting W is a path with t's edges by performing some sequence of probabilistic instructions

$L_{i_k j_k}(W_{s_k}, p_{i_k j_k})$ and passing successively some configurations with words c_k, $k = 1, 2, \ldots, t$, described in the following form:

$$q_0 W_1 W_2 \cdots W_m \overset{L_{i_1 j_1}(W_{s_1}, p_{i_1 j_1})}{\hookrightarrow_{\mathcal{M}}} c_1 \overset{L_{i_2 j_2}(W_{s_2}, p_{i_2 j_2})}{\hookrightarrow_{\mathcal{M}}} \cdots \overset{L_{i_{t-1} j_{t-1}}(W_{s_{t-1}}, p_{i_{t-1} j_{t-1}})}{\hookrightarrow_{\mathcal{M}}} c_{t-1} \overset{L_{i_t j_t}(W_{s_t}, p_{i_t j_t})}{\hookrightarrow_{\mathcal{M}}} c_t$$

where $c_t \in \mathcal{C}_{\mathrm{a}}(\mathcal{D}(\Sigma), \mathcal{M})$ is an accepting configuration with words, and the probability is

$$\prod_{k=1}^{t} Pr\left(c_{k-1} \overset{L_{i_k j_k}(W_{s_k}, p_{i_k j_k})}{\hookrightarrow_{\mathcal{M}}} c_k\right) = \prod_{k=1}^{t} p_{i_k j_k},$$

where $c_0 = q_0 W_1 W_2 \cdots W_m$.

Let $\hookrightarrow_{\mathcal{M}}^*$ denote the reflexive and transitive closure of $\hookrightarrow_{\mathcal{M}}$. Then $c_0 \hookrightarrow_{\mathcal{M}}^* c_{\mathrm{s}}$ represents the set of all allowable different paths from configurations with words c_0 to c_{s}, and its probability $Pr(c_0 \hookrightarrow_{\mathcal{M}}^* c_{\mathrm{s}})$ is the sum of those probabilities for all different computations (paths) from c_0 to c_{s}. For natural number t, $c_0 \hookrightarrow_{\mathcal{M}}^{*,t} c_{\mathrm{s}}$ stands for the set of all allowable paths from c_0 to c_{s} in exact ts steps. Therefore, the accepting and rejecting probabilities for computing string of words $W_1 W_2 \cdots W_m$ are defined, respectively, as:

$$Pr^{\mathrm{acc}}_{\mathcal{M}}(W_1 W_2 \cdots W_m) = \sum_{t=1}^{\infty} \sum_{c_{\mathrm{a}} \in \mathcal{C}_{\mathrm{a}}(\mathcal{D}(\Sigma), \mathcal{M})} Pr(q_0 W_1 W_2 \cdots W_m \hookrightarrow_{\mathcal{M}}^{*,t} c_{\mathrm{a}}), \tag{12}$$

$$Pr^{\mathrm{rej}}_{\mathcal{M}}(W_1 W_2 \cdots W_m) = \sum_{t=1}^{\infty} \sum_{c_{\mathrm{r}} \in \mathcal{C}_{\mathrm{r}}(\mathcal{D}(\Sigma), \mathcal{M})} Pr(q_0 W_1 W_2 \cdots W_m \hookrightarrow_{\mathcal{M}}^{*,t} c_{\mathrm{r}}), \tag{13}$$

where $Pr(q_0 W_1 W_2 \cdots W_m \hookrightarrow_{\mathcal{M}}^{*,t} c_{\mathrm{a}})$ represents the sum of probabilities for all computations of input $W_1 W_2 \cdots W_m$ halting in the same accepting configuration c_{a} in exact ts steps; the analogous explanation is suitable to $Pr(q_0 W_1 W_2 \cdots W_m \hookrightarrow_{\mathcal{M}}^{*,t} c_{\mathrm{r}})$.

Remark 4. As mentioned in Remark 3, on a standard PTM $\mathcal{M}$ with input alphabet Σ and starting state q_0, all computations for each given input $w \in \Sigma^*$ have the same number of computation steps, which is denoted by $N(w)$. A question naturally raised is that on a standard PTM, if all computations for each input string of words instead of symbols have the same number of computation steps? It may be worth considering, since this is also a new feature of PTMs with input strings of words.

From the viewpoint of computational complexity concerned with time measure, if for every input string of length n, PTM $\mathcal{M}$ makes at most $T(n)$ moves before arriving at all halting, accepting, or rejecting, and there is at least a computation for some string of length n whose number of steps is $T(n)$, then M is said to be *of time complexity* $T(n)$. If a PTM $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}})$, is *of time complexity* $T(n)$, then from Eqs. (10), (12) and (13) it is easy to see that when the input is some string $W_1 W_2 \cdots W_k$ of words, the number of computation steps likely exceed $|\Sigma|^k T(k)$, where notation $|X|$ denotes the number of elements in set X. Therefore, the time complexity for TM $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{a}}, q_{\mathrm{r}})$ with input strings of words, of time complexity $T(n)$, is not smaller than $|\Sigma|^n T(n)$.

As in the case of probabilistic automata (Theorem 1), we hope that the computations of $Pr^{\mathrm{acc}}_{\mathcal{M}}(W_1 W_2 \cdots W_m)$ and $Pr^{\mathrm{rej}}_{\mathcal{M}}(W_1 W_2 \cdots W_m)$ could be processed in terms of computing $Pr^{\mathrm{acc}}_{\mathcal{M}}(\sigma_1 \sigma_2 \cdots \sigma_m)$

and $Pr^{\mathrm{rej}}_{\mathcal{M}}(\sigma_1\sigma_2\cdots\sigma_m)$ for all possible $\sigma_i \in \Sigma$, $i = 1, 2, \ldots, m$; more exactly, we hope to have the following equalities called computational tractability as in Section 2:

$$Pr^{\mathrm{acc}}_{\mathcal{M}}(W_1W_2\cdots W_m) = \sum_{\sigma_1,\sigma_2,\ldots,\sigma_m\in\Sigma} Pr^{\mathrm{acc}}_{\mathcal{M}}(\sigma_1\sigma_2\cdots\sigma_m)\left(\prod_{i=1}^{m} W_i\right)(\sigma_1\sigma_2\cdots\sigma_m); \tag{14}$$

$$Pr^{\mathrm{rej}}_{\mathcal{M}}(W_1W_2\cdots W_m) = \sum_{\sigma_1,\sigma_2,\ldots,\sigma_m\in\Sigma} Pr^{\mathrm{rej}}_{\mathcal{M}}(\sigma_1\sigma_2\cdots\sigma_m)\left(\prod_{i=1}^{m} W_i\right)(\sigma_1\sigma_2\cdots\sigma_m). \tag{15}$$

Nonetheless, unfortunately, the above Eqs. (14) and (15) may not hold for some PTMs, and some input strings of words. Now we describe an example in detail to verify this conclusion, while the example also serves to show the formulation of PTMs together with related concepts defined above.

Example 2. Let $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{r}}, q_{\mathrm{a}})$ be a PTM, where $Q = \{q_0, q_1, q_2, q_{\mathrm{r}}, q_{\mathrm{a}}\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b, c, X, \#\}$, and δ is defined in terms of probabilistic instructions (Eq. (6)) as follows:

$L_{01}(0.5)$: $\langle q_0, a, q_1, \#, R\rangle$,
$L_{02}(0.5)$: $\langle q_0, a, q_1, X, R\rangle$,
$L_{03}(0.75)$: $\langle q_0, b, q_{\mathrm{a}}, \#, R\rangle$,
$L_{04}(0.25)$: $\langle q_0, b, q_{\mathrm{a}}, \#, R\rangle$,
$L_{11}(0.75)$: $\langle q_1, a, q_1, \#, R\rangle$,
$L_{12}(0.25)$: $\langle q_1, a, q_1, \#, L\rangle$,
$L_{13}(0.75)$: $\langle q_1, b, q_{\mathrm{r}}, \#, R\rangle$,
$L_{14}(0.25)$: $\langle q_1, b, q_{\mathrm{a}}, \#, L\rangle$,
$L_{21}(0.5)$: $\langle q_2, c, q_{\mathrm{a}}, \#, R\rangle$,
$L_{22}(0.5)$: $\langle q_2, a, q_2, \#, L\rangle$.

The computations for input strings *abc*, *aac*, *bac*, and *bbc* are depicted by the following Fig. 1 (a)–(d), respectively:

Note in above figures, those probabilities in the probabilistic instructions are omitted. In light of Fig. 1(a)–(d), we can calculate that

$$\begin{aligned}
&Pr^{\mathrm{rej}}_{\mathcal{M}}(aac) = 0, && Pr^{\mathrm{acc}}_{\mathcal{M}}(aac) = 0,\\
&Pr^{\mathrm{rej}}_{\mathcal{M}}(abc) = 0.75, && Pr^{\mathrm{acc}}_{\mathcal{M}}(abc) = 0.25,\\
&Pr^{\mathrm{rej}}_{\mathcal{M}}(bac) = 0, && Pr^{\mathrm{acc}}_{\mathcal{M}}(bac) = 1,\\
&Pr^{\mathrm{rej}}_{\mathcal{M}}(bbc) = 0, && Pr^{\mathrm{acc}}_{\mathcal{M}}(bbc) = 1
\end{aligned}$$

and, also, from Fig. 1(a)–(d) it is readily to derive that

$$\begin{aligned}
&Pr^{\mathrm{rej}}_{\mathcal{M}}(cxy) = 0, \ Pr^{\mathrm{acc}}_{\mathcal{M}}(cxy) = 0,\\
&Pr^{\mathrm{rej}}_{\mathcal{M}}(acx) = 0, \ Pr^{\mathrm{acc}}_{\mathcal{M}}(acx) = 0,\\
&Pr^{\mathrm{rej}}_{\mathcal{M}}(bac) = 0, \ Pr^{\mathrm{acc}}_{\mathcal{M}}(bac) = 1,\\
&Pr^{\mathrm{rej}}_{\mathcal{M}}(bcx) = 0, \ Pr^{\mathrm{acc}}_{\mathcal{M}}(bcx) = 1,
\end{aligned}$$

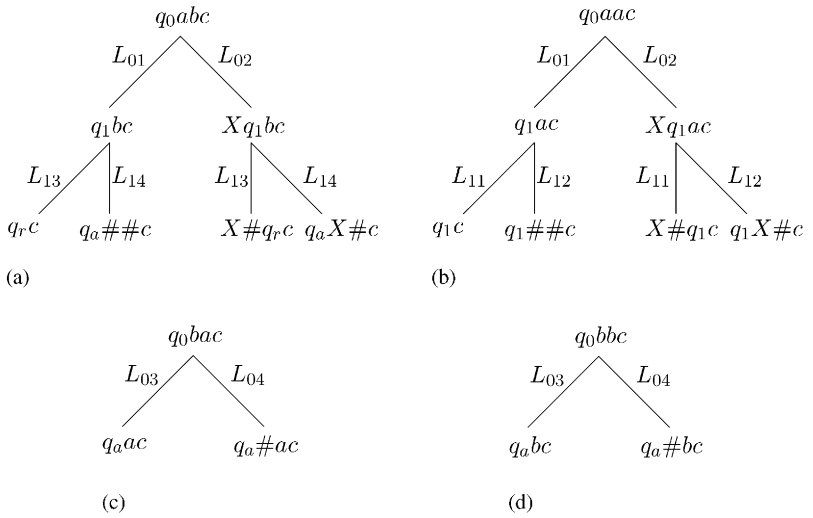

Fig. 1. (a) Computation tree of abc, (b) Computation tree of aac, (c) Computation tree of bac, (d) Computation tree of bbc.

for all $x, y \in \Sigma$. Set words W_1, W_2 (probability distributions over Σ) as:

$W_1 = 0.2\backslash a + 0.8\backslash b + 0\backslash c$,
$W_2 = 0.7\backslash a + 0.3\backslash b + 0\backslash c$,
$W_3 = 0.4\backslash a + 0.5\backslash b + 0.1\backslash c$.

Then

$$\begin{aligned}
&\sum_{\sigma_1,\sigma_2,\sigma_3\in\Sigma} Pr^{\mathrm{acc}}_{\mathcal{M}}(\sigma_1\sigma_2\sigma_3)W_1(\sigma_1)W_2(\sigma_2)\cdot W_3(\sigma_3)\\
&\quad = Pr^{\mathrm{acc}}_{\mathcal{M}}(abc)W_1(a)W_2(b) + Pr^{\mathrm{acc}}_{\mathcal{M}}(bac)W_1(b)W_2(a)\\
&\qquad + Pr^{\mathrm{acc}}_{\mathcal{M}}(bbc)W_1(b)W_2(b) + 0\\
&\quad = 0.25\times 0.2\times 0.3\times 0.1 + 0.8\times 0.7 + 0.8\times 0.3\times 0.1\\
&\quad = 0.0815.
\end{aligned}$$

On the other hand, by means of Eqs. (10) and (11) some of probabilistic instructions (with words) that will be used are as follows:

$$\begin{aligned}
&L_{01}(W_1, 0.1): \quad \langle q_0, W_1, q_1, \#, R\rangle,\\
&L_{02}(W_1, 0.1): \quad \langle q_0, W_1, q_1, X, R\rangle,\\
&L_{03}(W_1, 0.6): \quad \langle q_0, W_1, q_{\mathrm{a}}, \#, R\rangle,\\
&L_{04}(W_1, 0.2): \quad \langle q_0, W_1, q_{\mathrm{a}}, \#, L\rangle,\\
&L_{11}(W_2, 0.525): \langle q_1, W_2, q_1, \#, R\rangle,\\
&L_{12}(W_2, 0.175): \langle q_1, W_2, q_1, \#, L\rangle,
\end{aligned}$$

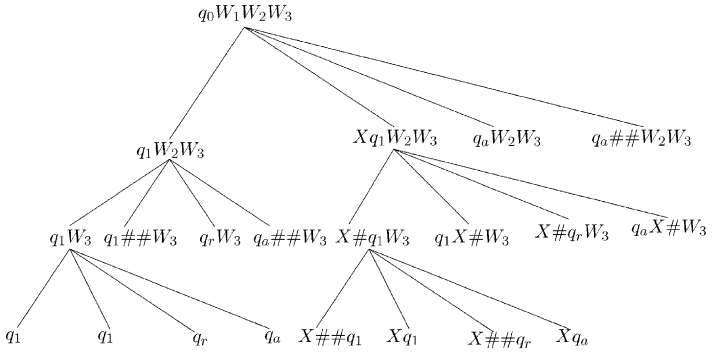

Fig. 2. Computation tree of $W_1W_2W_3$.

$$
\begin{aligned}
&L_{13}(W_2, 0.225) : \langle q_1, W_2, q_{\mathrm{r}}, \#, R\rangle,\\
&L_{14}(W_2, 0.075) : \langle q_1, W_2, q_{\mathrm{a}}, \#, L\rangle,\\
&L_{11}(W_3, 0.3) : \quad \langle q_1, W_3, q_1, \#, R\rangle,\\
&L_{12}(W_3, 0.1) : \quad \langle q_1, W_3, q_1, \#, L\rangle,\\
&L_{13}(W_3, 0.375) : \langle q_1, W_3, q_{\mathrm{r}}, \#, R\rangle,\\
&L_{14}(W_3, 0.125) : \langle q_1, W_3, q_{\mathrm{a}}, \#, L\rangle.
\end{aligned}
$$

The computation for input string of words $W_1W_2W_3$ is visualized by Fig. 2 as follows.

By combining Fig. 2 we obtain that

$$
\begin{aligned}
&Pr^{\mathrm{acc}}_{\mathcal{M}}(W_1W_2W_3)\\
&= \sum_{t=1}^{\infty} \sum_{c_{\mathrm{a}}\in\mathcal{C}_{\mathrm{a}}(\mathcal{D}(\Gamma),\mathcal{M})} Pr(q_0W_1W_2W_3 \hookrightarrow^{*,t}_{\mathcal{M}} C_{\mathrm{a}})\\
&= Pr(q_0W_1W_2W_3 \overset{L_{03}(W_1,0.6)}{\hookrightarrow_{\mathcal{M}}} q_{\mathrm{a}}W_2W_3)\\
&\quad + Pr(q_0W_1W_2W_3 \overset{L_{04}(W_1,0.2)}{\hookrightarrow_{\mathcal{M}}} q_{\mathrm{a}}\#W_2W_3)\\
&\quad + Pr(q_0W_1W_2W_3 \overset{L_{01}(W_1,0.1)}{\hookrightarrow_{\mathcal{M}}} q_1W_2W_3 \overset{L_{14}(W_2,0.075)}{\hookrightarrow_{\mathcal{M}}} q_{\mathrm{a}})\\
&\quad + Pr(q_0W_1W_2W_3 \overset{L_{02}(W_1,0.1)}{\hookrightarrow_{\mathcal{M}}} Xq_1W_2W_3 \overset{L_{14}(W_2,0.075)}{\hookrightarrow_{\mathcal{M}}} q_{\mathrm{a}}X\#)\\
&\quad + Pr(q_0W_1W_2W_3 \overset{L_{01}(W_1,0.1)}{\hookrightarrow_{\mathcal{M}}} q_1W_2W_3 \overset{L_{11}(W_2,0.525)}{\hookrightarrow_{\mathcal{M}}} q_1W_3 \overset{L_{14}(W_3,0.125)}{\hookrightarrow_{\mathcal{M}}} q_{\mathrm{a}})\\
&\quad + Pr(q_0W_1W_2W_3 \overset{L_{02}(W_1,0.1)}{\hookrightarrow_{\mathcal{M}}} Xq_1W_2W_3 \overset{L_{11}(W_2,0.525)}{\hookrightarrow_{\mathcal{M}}} X\#q_1W_3 \overset{L_{14}(W_3,0.125)}{\hookrightarrow_{\mathcal{M}}} Xq_{\mathrm{a}})\\
&= 0.828125.
\end{aligned}
$$

So, we show that

$$
Pr^{\mathrm{acc}}_{\mathcal{M}}(W_1W_2W_3) \neq \sum_{\sigma_1,\sigma_2,\sigma_3\in\Sigma} Pr^{\mathrm{acc}}_{\mathcal{M}}(\sigma_1\sigma_2\sigma_3)W_1(\sigma_1)\cdot W_2(\sigma_2)\cdot W_3(\sigma_3),
$$

which shows that Eq. (14) may not hold. If exchanging the accepting and rejecting states in above PTM, then it also shows that Eq. (15) does not hold.

Remark 5. Example 2 shows that Eqs. (14) and (15) may not hold, which implies an essential difference of PTMs from PFAs since analogous conclusion does hold for PFAs (Theorem 1). We may see this result by intuition. Indeed, in the right-hand formula of Eq. (12), the machine M, before some inputted words not being scanned, may attains a final accepted configuration, whereas, in the right-hand ones of Eq. (14), we know that all these inputted words likely have a certain influence on the value, so it is possible to make the right-hand formulae of Eqs. (12) and (14) unequal by taking appropriate words W_i. Similar analysis can be applied to Eqs. (13) and (15). Exactly, Example 2 has verified these observations. Naturally, We may ask under what conditions Eqs. (14) and (15) hold true?

4. Probabilistic context-free grammars with input strings of words

In this section, we will deal with PCFGs and PRGs with input strings of words. First let us recall briefly several conventional grammars.

In general, a grammar is denoted by a quadruple $G = (V, T, P, S)$, where V and T are finite sets of *variables* and *terminals*, respectively, with $V \cap T = \emptyset$; P is a finite set of productions, each of which is of the form $\alpha \to \beta$ where α, β are strings of symbols from $(V \cup T)^*$, and $\alpha \neq \beta$, S in V is a special variable called the start symbol. Such a grammar is unrestricted on P called *type* 0 *grammar* that produces the same languages as those by Turing machines, that is, *recursively enumerable* languages. In particular, some constraints imposed on P follow the classification of grammars as follows:

Type 1 *grammar* (*context-sensitive*): The productions are restricted as $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ where $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$, $A \in V$, $\beta \neq \varepsilon$.

Type 2 *grammar* (*context-free*). The permissible productions are the forms of $A \to \beta$, where $A \in V$, $\beta \in (V \cup T)^*$ with $\beta \neq \varepsilon$. Type 2 grammars have the same production power as non-deterministic pushdown automata, and, as was known, they generate context-free languages.

Type 3 *grammar* (*regular*). The allowable productions are of the forms $A \to aB$ or $A \to a$, where $a \in T$, $A, B \in V$. Type 3 grammars, i.e., regular grammars familiar to us, are exactly equivalent to finite-state automata, generating regular languages.

Formally to define the language generated by a grammar $G = (V, T, P, S)$, we first define two relations $\Rightarrow_G$ and $\Rightarrow_G^*$ between strings in $(V \cup T)^*$. If $\alpha \to \beta \in P$, then $\gamma_1 \alpha \gamma_2 \Rightarrow_G \gamma_1 \beta \gamma_2$ for any $\gamma_1, \gamma_2 \in (V \cup T)^*$, and, we call that $\gamma_1 \alpha \gamma_2$ *derives directly* $\gamma_1 \beta \gamma_2$ in grammar G. $\Rightarrow_G^*$ is the reflexive and transitive closure of $\Rightarrow_G$. For simplicity, we use $\Rightarrow$ and $\Rightarrow^*$ for $\Rightarrow_G$ and $\Rightarrow_G^*$, respectively, if there is no confusion. More specifically, if $\alpha_1, \alpha_2, \ldots, \alpha_m \in (V \cup T)^*$, and, $\alpha_i \Rightarrow \alpha_{i+1}$, $i = 1, 2, \ldots, m-1$, then we say $\alpha_1 \Rightarrow^* \alpha_m$ or α_1 derives α_m in grammar G.

The *language* generated by G is a subset of T^*, that is,

$$\{w : w \in T^*, S \Rightarrow^* w\}.$$

Definition 3. A probabilistic grammar is a system $G = (V, T, P, S)$, where V, T, P and S are the same as *type* 0 *grammar* above, but the expressions for the form of productions $\alpha_i \to \beta_j$ are endowed with certain probability as $p_{ij}(\alpha_i \to \beta_j) \in (0, 1]$, representing the probability of α_i producing β_j, where

p_{ij} like L_{ij} in *probabilistic program lines* is just a mark standing for certain production. Furthermore, if $\alpha_i \to \beta_j \in P$, then for all those productions of the form with α_i in left side, $\alpha_i \to \beta_j$ $(1 \leqslant j \leqslant k_i)$ satisfying

$$\sum_{j=1}^{k_i} p_{ij}(\alpha_i \to \beta_j) = 1, \tag{16}$$

where notably $\beta_j \neq \alpha_i$ as indicated above.

For convenience, we directly represent a production with the form of $p_{ij}(\alpha_i \to \beta_j)$. For $(\alpha, \beta) \in (V \cup T)^* \times (V \cup T)^*$, we use $\alpha \overset{p_{ij}}{\Rightarrow} \beta$ to stand for α *directly deriving* β by applying a production $p_{ij}(\alpha_i \to \beta_j)$ in P.

Generally, a derivation from α to β consists of some direct derivations, represented as

$$\alpha \overset{p_{i_1 j_1}}{\Rightarrow} \alpha_1 \overset{p_{i_2 j_2}}{\Rightarrow} \ldots \overset{p_{i_k j_k}}{\Rightarrow} \beta, \tag{17}$$

in which the productions $p_{i_s j_s}$ $(s = 1, 2, \ldots, k)$ are successively used, and

$$\{p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_k j_k}\}$$

may be a multi-set, [1] and the probability is $\prod_{s=1}^{k} p_{i_s j_s}$.

In this section, we focus mainly on PCFGs and PRGs for inputting strings of words.

Definition 4. A PCFG is a system $G = (V, T, P, S)$, where V, T, and S are the same as above, but the productions are only the form

$$p_{ij}(A_i \to \beta_j) \tag{18}$$

where $A_i \in V$ and $\beta_j \in (V \cup T)^*$.

Definition 5. A PRG is a system $G = (V, T, P, S)$, where V, T, and S are the same as above, but the productions are only the form

$$p_{ij}(A_i \to aA_j), \tag{19}$$

where $a \in T$, $A_i \in V$, and $A_j \in V \cup \{\varepsilon\}$.

Remark 6. Two simplification types of CFGs are Chomsky normal form (CNF) and Greibach normal form (GNF) [17]. Concerning PCFGs, the equivalences still hold [12, Theorems 6.2 and 6.3]. More specifically, a PCFG $G = (V, T, P, S)$ is of Chomsky normal form, if the productions are of the form $p_{ij}(A_i \to A_j A_k)$ or $p_{ij}(A_i \to a_j)$ where $A_i, A_j, A_k \in V$, and $a_j \in T$; G is of Greibach normal form if the productions are of the form $p_{ij}(A_i \to a_j \beta_j)$ where $a_j \in T$, $A_i \in V$ and $\beta_j \in V^*$. In light of [12] it was verified that every PCFG is equivalent to a CNF and a GNF.

[1] A multi-set allows repeated elements, for example, $\{1, 1, 2\}$ is a multi-set, and $\{1, 1, 2\} \neq \{1, 2\}$.

For the sake of generality, we here consider general PCFGs with input strings of words, and therefore demonstrate that each PCFG is equivalent to a CNF and a GNF, in the sense that the probability for generating each string of *words* is equal.

For any $\alpha \in (V \cup T)^*$ and $A_i \in V$, a derivation from A_i to α is the form

$$A_i \overset{p_{i_1 j_1}}{\Rightarrow} \alpha_1 \overset{p_{i_2 j_2}}{\Rightarrow} \dots \overset{p_{i_k j_k}}{\Rightarrow} \alpha, \tag{20}$$

which can be equivalently depicted by a deriving tree (see [17] for example). It is worth noting that two derivations from A_i to α are different if and only if they have distinct trees of derivation, and equivalently, their sets of productions used are unequal, which is able to be equivalently characterized by *leftmost derivation*, as well. A derivation from A_i to α is called to be *leftmost*, if every direct derivation in the derivation rewrites the leftmost variable. For instance, in Eq. (20), α_1 and α_2 satisfies that $\alpha_1 = wA_{i_2}\beta_1 \overset{p_{i_2 j_2}}{\Rightarrow} w\beta_{j_2}\beta_1 = \alpha_{i+1}$ where $w \in T^*$, $\beta_1 \in (V \cup T)^*$, and $p_{i_2 j_2}(A_{i_2} \to \beta_{j_2}) \in P$. For any $A_i \in V$, and $\alpha \in (V \cup T)^*$, in light of [17,33], the number of different derivations from A_i to α is equal to that of different *leftmost* derivations from A_i to α.

Now we consider PCFGs $G = (V, T, P, S)$. Suppose that the set of words to be computed is as

$$\mathcal{W} = \{W_1, W_2, \dots, W_m\}.$$

For any words $W_i \in \mathcal{W}$ $(i = 1, 2, \dots, m)$, and any production $p_{ij}(A_i \to \beta_j) \in P$ where we assume $\beta_j = a_1\gamma_1 a_2\gamma_2 \cdots a_k\gamma_k$, $a_j \in T$, and $\gamma_j \in V^*$ $(j = 1, 2, \dots, k)$, then we can generalize the *productions* to *productions with words* in the following manner:

$$\begin{aligned} &p_{il}^{\mathcal{W}}(A_i \to W_1\gamma_1 W_2\gamma_2 \cdots W_k) \\ &\quad = \sum_{t_1, t_2, \dots, t_k \in T} p_{ij_s}(A_i \to t_1\gamma_1 t_2\gamma_2 \cdots t_k\gamma_k) \times \left(\prod_{i=1}^{k} W_i\right)(t_1 t_2 \cdots t_k), \end{aligned} \tag{21}$$

where the subscript j_s in p_{ij_s} corresponds to $\beta_{j_s} = t_1\gamma_1 t_2\gamma_2 \cdots t_k\gamma_k$ for some $t_1, t_2, \dots, t_k \in T$, and due to Eq. (18) there is at least j corresponding to $\beta_j = a_1\gamma_1 a_2\gamma_2 \cdots a_k\gamma_k$, $l = 1, 2, \dots, l(p_{ij}, \mathcal{W})$, and $l(p_{ij}, \mathcal{W})$ is the number of different productions with words yielded by production p_{ij} (Eq. (18)) together with $\mathcal{W}$. We now provide an example to illustrate the formulation of these concepts, and a more detailed one is referred to Example 4.

Example 3. Let $G = (V, T, P, S)$ be a PCFG, where $V = \{A_1 = S, A_2\}$, $T = \{a, b\}$, and suppose $p_{11}(A_1 \to aA_2b) = \frac{1}{4}$, $p_{12}(A_1 \to bW_2a) = \frac{3}{4} \in P$. Assume that the set of words computed is as

$$\mathcal{W} = \{W_1, W_2\},$$

where $W_1 = 0.1\backslash a + 0.9\backslash b$, $W_2 = 0.7\backslash a + 0.3\backslash b$. Then by Eq. (21) it yields the following productions with words:

$$p_{11}^{\mathcal{W}}(A_1 \to W_1A_2W_1) = 0.09,\ p_{12}^{\mathcal{W}}(A_1 \to W_1A_2W_2) = 0.48,$$

$$p_{13}^{\mathcal{W}}(A_1 \to W_2A_2W_1) = 0.48,\ p_{14}^{\mathcal{W}}(A_1 \to W_2A_2W_2) = 0.21.$$

We denote by $P^{\mathcal{W}}$ the set of productions with words for PCFG G defined above.

Binary relation $\Rightarrow$ over $(V \cup T)^*$ can be extended to $(\mathcal{D}(T) \cup V)^*$. For any $\alpha, \beta \in (\mathcal{D}(T) \cup V)^*$, and $p_{il}^{\mathcal{W}}(A_i \to W_1\gamma_1 W_2\gamma_2 \cdots W_k\gamma_k) \in P^{\mathcal{W}}$, then $\alpha A_i\beta \overset{p_{il}^{\mathcal{W}}}{\Rightarrow} \alpha A_1\gamma_1 W_2\gamma_2 \cdots W_k\gamma_k\beta$ represents that $\alpha A_i\beta$ *derives directly with words* $\alpha A_1\gamma_1 W_2\gamma_2 \cdots W_k\gamma_k\beta$ in one step by applying production with words $p_{il}^{\mathcal{W}}(A_i \to W_1\gamma_1 W_2\gamma_2 \cdots W_k) \in P^{\mathcal{W}}$, and the corresponding probability is naturally as:

$$Pr(\alpha A_i\beta \overset{p_{il}^{\mathcal{W}}}{\Rightarrow} \alpha A_1\gamma_1 W_2\gamma_2 \cdots W_k\gamma_k\beta) = p_{il}^{\mathcal{W}}(A_i \to W_1\gamma_1 W_2\gamma_2 \cdots W_k\gamma_k).$$

For any $A_i \in V$, and $\alpha \in (\mathcal{D}(T) \cup V)^*$, a *derivation with words* from A_i to α is represented as

$$\Delta(A_i, \alpha) : A_i \overset{p_{i_1 j_1}^{\mathcal{W}}}{\Rightarrow} \alpha_1 \overset{p_{i_2 j_2}^{\mathcal{W}}}{\Rightarrow} \ldots \overset{p_{i_k j_k}^{\mathcal{W}}}{\Rightarrow} W \tag{22}$$

for some natural number k and $p_{i_l j_l}^{\mathcal{W}} \in P^{\mathcal{W}}$, $k = 1, 2, \ldots, k$, and the probability denoted by $Pr(\Delta(A_i, \alpha))$, is as

$$Pr(\Delta(A_i, \alpha)) = \prod_{l=1}^{k} p_{i_l j_l}^{\mathcal{W}},$$

in which, notably, the productions with words used may be a multi-set, denoted by $P(\Delta(A_i, \alpha))$,

$$P(\Delta(A_i, \alpha)) = \{p_{i_1 j_1}^{\mathcal{W}}, p_{i_2 j_2}^{\mathcal{W}}, \ldots, p_{i_k j_k}^{\mathcal{W}}\}. \tag{23}$$

Two derivations with words $\Delta_1(A_i, \alpha)$ and $\Delta_2(A_i, \alpha)$ are different if and only if

$$P(\Delta_1(A_i, \alpha)) \neq P(\Delta_2(A_i, \alpha)).$$

To define the probability of A_i deriving W, we further recall *leftmost derivation* (or called *left canonical derivation* in terms of [33], which is exactly the same as the *leftmost* derivation defined above, that is, every direct derivation with words rewrites the leftmost variable.

Remark 7. As pointed out above, by virtue of [17] or Theorem 2 in [33], the number of distinct derivations with words from S to W is exactly equal to the number of different leftmost derivations with words from S to W. It is worth indicating that though this conclusion in [17] and [33] is in the case of the generated strings to be those strings of terminals instead of words, there is without any essential difference for the case of strings of words at this point.

For convenience, we denote by $D(A_i, \alpha)$ the set of all different leftmost derivations with words from A_i to α. In particular, $D(A_i, w)$ represents the set of all leftmost derivations from A_i to w in the usual way, where $w \in T^*$ is a string of symbols. So, the probability of A_i deriving with words to α, denoted by $Pr(A_i \Rightarrow^* \alpha)$, is defined as the sum of those probabilities for all distinct leftmost derivations with words, that is,

$$Pr(A_i \Rightarrow^* \alpha) = \sum\{Pr(\Delta(A_i, \alpha)) : \Delta(A_i, \alpha) \in D(A_i, \alpha)\}. \tag{24}$$

Therefore, for any string of words $W = W_1 W_2 \cdots W_k \in \mathcal{D}(T)^*$, the probability of W being generated by PCFG G, denoted by $Pr_G(W)$, is as:

$$Pr_G(W) = Pr(S \Rightarrow^* W). \tag{25}$$

Similar to Theorem 1, we hope to calculate $Pr_G(W)$ by means of computing $Pr_G(t_1t_2\cdots t_k)$ for all $t_1, t_2, \ldots, t_k \in T$. Actually, we have the following theorem.

Theorem 2. *Let $G = (V, T, P, S)$ be a* PCFG, *and $W = W_1W_2\cdots W_m \in \mathcal{D}(T)^*$. Then*

$$Pr_G(W) = \sum_{t_1,t_2,\ldots,t_m \in T} Pr_G(t_1t_2\cdots t_m) \times \left(\prod_{i=1}^{m} W_i\right)(t_1t_2\cdots t_m). \tag{26}$$

Before proving this theorem, we first need to give two notations. Let $t = t_1t_2\cdots t_m \in T^*$ and $W = W_1W_2\cdots W_m \in \mathcal{D}(T)^*$, where $t_i \in T$, $W_i \in \mathcal{D}(T)$, $i = 1, 2, \ldots, m$. If $\alpha(t) = \alpha_1 t_{i_1}\alpha_2 t_{i_2}\cdots t_{i_{k-1}}\alpha_k \in (V \cup T)^*$ where $t_{i_l} \in \{t_1, t_2, \ldots, t_m\}$ for $l = 1, 2, \ldots, k-1$, and $\alpha_j \in V^*$, $j = 1, 2, \ldots, k$; and $\alpha(W) = \alpha_1 W_{i_1}\alpha_2 W_{i_2}\cdots W_{i_{k-1}}\alpha_k \in (V \cup \mathcal{D}(T))^*$ where $W_{i_l} \in \{W_1, W_2, \ldots, W_m\}$, $l = 1, 2, \ldots, k-1$. Then, we call $\alpha(t)$ and $\alpha(W)$ to be (t, W)-*equivalent*, and denote it by $\alpha(t) \sim \alpha(W)$.

For any $\Delta(S, W) \in D(S, W)$, where $W = W_1W_2\cdots W_m \in \mathcal{D}(T)^*$, it is, for example, as follows:

$$\Delta(S, W) : S \overset{p^{\mathcal{W}}_{i_1j_1}}{\Rightarrow} \alpha_1(W) \overset{p^{\mathcal{W}}_{i_2j_2}}{\Rightarrow} \ldots \overset{p^{\mathcal{W}}_{i_{k-1}j_{k-1}}}{\Rightarrow} \alpha_{k-1}(W) \overset{p^{\mathcal{W}}_{i_kj_k}}{\Rightarrow} W, \tag{27}$$

where $p^{\mathcal{W}}_{i_lj_l}(A_{i_l} \to \beta_{j_l}(W))$ $(l = 1, 2, \ldots, k)$ are productions with words, and $\beta_{j_l}(W) \in (\mathcal{D}(T) \cup V)^*$, and it is easy to know that $p^{\mathcal{W}}_{i_lj_l}$ is yielded from some production of the form $p_{i_ls}(A_{i_l} \to \beta_{j_l}(t))$ for some $t = t_1t_2\cdots t_m \in T^*$, where $\beta_{j_l}(t) \sim \beta_{j_l}(W)$. Now for any $t = t_1t_2\cdots t_m \in T^*$, if there exist s_l $(l = 1, 2, \ldots, k)$ such that $p_{i_ls_l}(A_{i_l} \to \beta_{j_l}(t)) \in P$, then we obtain a leftmost derivation from S to t, $\Delta(S, t)$, and denote it by $\Delta(S, t) \asymp \Delta(S, W)$. Indeed, there is a 1-1 correspondence between $D(S, W)$ and $D(S, t)$ established in this manner.

Lemma 1. *For any $W = W_1W_2\cdots W_m \in \mathcal{D}(T)^*$,*

$$Pr(\Delta(S, W)) = \sum\{Pr(\Delta(S, t)) \times \prod_{i=1}^{m} W_i(t_i) : t = t_1t_2\cdots t_m \in T^*, \Delta(S, t) \asymp \Delta(S, W)\}.$$

Proof. Assume that $\Delta(S, W)$ is described by Eq. (27). Set

$$\mathcal{J}(\Delta(S, W)) = \{t = t_1t_2\cdots t_m : t_i \in T, i = 1, 2, \ldots, m, \text{ there are } s_l, \text{ such that } p_{i_ls_l}(A_{i_l} \to \beta_{i_l}(t)) \in P, l = 1, 2, \ldots, k\}.$$

In light of the definition of $p^{\mathcal{W}}_{i_lj_l}(A_{i_l} \to \beta_{j_l}(W))$, with Eq. (27) we have

$$\begin{aligned} Pr(\Delta(S, W)) &= \prod_{k=1}^{k} p^{\mathcal{W}}_{i_lj_l}(A_{i_l} \to \beta_{j_l}(W)) \\ &= \sum_{t=t_1t_2\cdots t_m \in \mathcal{J}(\Delta(S,W))} \prod_{l=1}^{k} p_{i_ls_l}(A_{i_l} \to \beta_{j_l}(t)) \prod_{i=1}^{m} W_i(t_i) + \sum_{t \notin \mathcal{J}(\Delta(S,W))} 0 \\ &= \sum_{t=t_1t_2\cdots t_m} \{Pr(\Delta(S, t)) \prod_{i=1}^{m} W_i(t_i) : \Delta(S, t) \asymp \Delta(S, W)\}. \qquad \square \end{aligned}$$

Lemma 2. *For any* $t = t_1 t_2 \cdots t_m \in T^*$ *and any* $W = W_1 W_2 \cdots W_m \in \mathcal{D}(T)^*$,

$$D(S,t) = \{\Delta(S,t) : \Delta(S,t) \asymp \Delta(S,W), \Delta(S,W) \in D(S,W)\}.$$

Proof. Suppose $\Delta(S,t) \in D(S,t)$, and $\Delta(S,t)$ is as follows:

$$\Delta(S,t) : S \overset{p_{i_1 j_1}}{\Rightarrow} \alpha_1(t) \overset{p_{i_2 j_2}}{\Rightarrow} \cdots \overset{p_{i_{k-1} j_{k-1}}}{\Rightarrow} \alpha_{k-1}(t) \overset{p_{i_k j_k}}{\Rightarrow} t,$$

for some $p_{i_l j_l}(A_{i_l} \to \beta_{j_l}(t)) \in P$, $l = 1, 2, \ldots, k$. By replacing $\beta_{j_l}(t)$ and $\alpha_l(t)$ with $\beta_{j_l}(W)$ and $\alpha_l(W)$, respectively, then $\Delta(S,W) \in D(S,W)$ and $\Delta(S,W) \asymp \Delta(S,t)$, so Lemma 2 is proved. □

Proof of Theorem 2. With Lemmas 1 and 2, we have that

$$\begin{aligned}
Pr_G(W) &= \sum \{Pr(\Delta(S,W)) : \Delta(S,W) \in D(S,W)\} \\
&= \sum_{\Delta(S,W) \in D(S,W)} \sum \left\{ Pr(\Delta(S,t)) \prod_{i=1}^{m} W_i(t_i) : t = t_1 t_2 \cdots t_m \in T^*, \Delta(S,t) \asymp \Delta(S,W) \right\} \\
&= \sum_{t = t_1 t_2 \cdots t_m \in T^*} \sum \left\{ Pr(\Delta(S,t)) \prod_{i=1}^{m} W_i(t_i) : \Delta(S,t) \asymp \Delta(S,W), \Delta(S,W) \in D(S,W) \right\} \\
&= \sum_{t = t_1 t_2 \cdots t_m \in T^*} \prod_{i=1}^{m} W_i(t_i) \sum \{Pr(\Delta(S,t)) : \Delta(S,t) \in D(S,t)\}. \qquad \square
\end{aligned}$$

Because PRGs, Chomsky PCFGs, and Greibach PCFGs are also special PCFGs, we have:

Corollary 1. *Let* $G = (V, T, P, S)$ *be a* PRG, *Chomsky* PCFGs, *or Greibach* PCFGs. *For any* $W = W_1 W_2 \cdots W_m \in \mathcal{D}(T)^*$, *then*

$$Pr_G(W) = \sum_{t_1, t_2, \ldots, t_m \in T} Pr_G(t_1 t_2 \cdots t_m) \times \left(\prod_{i=1}^{m} W_i \right)(t_1 t_2 \cdots t_m). \tag{28}$$

Now we discuss a characterization of PRGs in terms of PFAs and the equivalence between a PCFG and its CNF or GNF. We need two results mentioned in Remark 6, which was shown by Fu [12]:

Theorem 3 (*Fu [12]*). (1) *For any* PRG $G = (V, T, P, S)$, *there exists a* PFA $\mathcal{A}$ *with the input alphabet* $\Sigma = T$ *such that for any* $w \in T^*$,

$$Pr_G(w) = Pr_{\mathcal{A}}(w). \tag{29}$$

(2) *For any* PCFG $G = (V, T, P, S)$, *there are a Chomsky* PCFG G_1 *and a Greibach* PCFG G_2 *with the same set of terminals T, such that for any* $w \in T^*$,

$$Pr_G(w) = Pr_{G_1}(w) = Pr_{G_2}(w). \tag{30}$$

Corollary 2. (1) *For any* PRG $G = (V, T, P, S)$, *there are a* PFA $\mathcal{A}$ *with the input alphabet* $\Sigma = T$ *such that for any* $W \in \mathcal{D}(T)^*$,

$$Pr_G(W) = Pr_{\mathcal{A}}(W). \tag{31}$$

(2) *For any* PCFG $G = (V, T, P, S)$, *there are a Chomsky* PCFG G_1 *and a Greibach* PCFG G_2 *with the same set of terminals T, such that for any* $W \in \mathcal{D}(T)^*$,

$$Pr_G(W) = Pr_{G_1}(W) = Pr_{G_2}(W). \tag{32}$$

Proof. The proof of (1) is direct from Theorem 1, Theorem 3 (1), and Corollary 1; the proof of (2) is straightforward by Theorems 2, 3 (2), and Corollary 1. □

In practice, as pointed out in the literature (see [33, p. 368, lines 11–15], for instance), most grammars are *unambiguous*; that is, there is only one leftmost derivation from start symbol to each string belonging to the language generated by an unambiguous grammar. Furthermore, similar to the issue of computational complexity, the expected derivation length denoted by $\mathrm{EDL}(A)$ of a derivation beginning with nonterminal A, which was defined in [33], is the expected number of steps in a derivation beginning in A and ending with a terminal string. From the preceding discussion it is clear to see that the expected number of steps in a derivation ending with strings of words are without essential change, which is to a certain extent different from the issue of computation complexity of PTMs with input strings of words.

Remark 8. A *probabilistic language* over an alphabet T is a pair $\langle L, f\rangle$ where L is a language generated by a conventional grammar with terminal set T, and f is a function from T^* to $[0, 1]$ satisfying: $f(x) = 0$ if $x \notin L$; and $\sum_{x\in L} f(x) = 1$. A probabilistic context-free language over alphabet T is a function $Pr_G : T^* \to [0, 1]$ for some PCFG G defined above. As is well-known, $L = \{a^n b^n : n \geqslant 0\}$ is a context-free language [17]. If the function ϕ over $\{a, b\}^*$ is defined as $\phi(a^n b^n) = \frac{1}{e\cdot n!}$, then $\langle L, \phi\rangle$ is a probabilistic language. However, in [6], Booth and Thompson showed that ϕ is not a probabilistic context-free language, that is to say, there is no PCFG G with terminal set $\{a, b\}$ satisfying: $Pr_G = \phi$. Here, we may raise whether there is a PCFG G with terminal set $\{a, b\}$ and some words over the terminal set (i.e., probability distributions), say W_1, W_2, such that $Pr_G(x) = 0$ for $x \notin L$, and $Pr_G(W_1^n W_2^n) = \frac{1}{e\cdot n!}$? We take it into account as an interesting problem and leave it open here.

As an illustration for the formulation of PCFGs for inputting strings of words, we describe an example to close this section.

Example 4. Let $G = (V, T, P, S)$ be a PCFG, where $V = \{A_1, A_2\}$, $S = A_1$ is the starting symbol, $T = \{a, b, c\}$, and P consists of the following probability productions:

$$p_{11}(A_1 \to aA_1A_1A_2) = 0.2;$$
$$p_{12}(A_1 \to A_2) = 0.8;$$
$$p_{21}(A_2 \to bA_1) = 0.6;$$
$$p_{22}(A_2 \to c) = 0.4.$$

Suppose that the set of words $\mathcal{W} = \{W_1, W_2\}$ is defined as follows:
$W_1 = 0.1\backslash a + 0.5\backslash b + 0.4\backslash c$;
$W_2 = 0.4\backslash a + 0.3\backslash b + 0.3\backslash c$.
Then in terms of Eq. (21), $P^{\mathcal{W}}$, the set of productions with words, is derived as follows:

$$p_{11}^{\mathcal{W}}(A_1 \to W_1A_1A_1A_2) = p_{11}(A_1 \to aA_1A_1A_2)W_1(a) = 0.02;$$
$$p_{12}^{\mathcal{W}}(A_1 \to W_2A_1A_1A_2) = p_{11}(A_1 \to aA_1A_1A_2)W_2(a) = 0.08;$$
$$p_{13}^{\mathcal{W}}(A_1 \to A_2) = p_{12}(A_1 \to A_2) = 0.8;$$
$$p_{21}^{\mathcal{W}}(A_2 \to W_1A_1) = p_{21}(A_2 \to bA_1)W_1(b) = 0.3;$$
$$p_{22}^{\mathcal{W}}(A_2 \to W_2A_1) = p_{21}(A_2 \to bA_1)W_2(b) = 0.18;$$
$$p_{23}^{\mathcal{W}}(A_2 \to W_1) = p_{22}(A_2 \to c)W_1(c) = 0.16;$$
$$p_{24}^{\mathcal{W}}(A_2 \to W_2) = p_{22}(A_2 \to c)W_2(c) = 0.12.$$

Then

$$\begin{aligned} Pr_G(bc) &= Pr(A_1 \Rightarrow A_2)Pr(A_2 \Rightarrow bA_1)Pr(bA_1 \Rightarrow bA_2)Pr(bA_2 \Rightarrow bc) \\ &= 0.8 \times 0.6 \times 0.8 \times 0.4 = 0.1536 \end{aligned}$$

and $Pr_G(ab) = Pr_G(ac) = 0$. By virtue of Theorem 2, we have

$$Pr_G(W_1W_2) = Pr_G(bc)W_1(b)W_2(c) = 0.1536 \times 0.5 \times 0.3 = 0.02304.$$

As well, we can directly compute $Pr_G(W_1W_2)$:

$$\begin{aligned} Pr_G(W_1W_2) &= Pr(A_1 \Rightarrow A_2)Pr(A_2 \Rightarrow W_1A_1)Pr(W_1A_1 \Rightarrow W_1A_2Pr(W_1A_2 \Rightarrow W_1W_2) \\ &= 0.8 \times 0.3 \times 0.8 \times 0.12 = 0.02304. \end{aligned}$$

$$\begin{aligned} Pr_G(accc) &= Pr(A_1 \Rightarrow aA_1A_1A_2)Pr(aA_1A_1A_2 \Rightarrow aA_2A_1A_2)Pr(aA_2A_1A_2 \Rightarrow aA_2A_2A_2) \\ &\quad \times Pr(aA_2A_2A_2 \Rightarrow acA_2A_2)Pr(acA_2A_2 \Rightarrow accA_2)Pr(accA_2 \Rightarrow accc) \\ &= 0.2 \times 0.8 \times 0.8 \times 0.4 \times 0.4 \times 0.4 = 0.008192 \end{aligned}$$

and it is easy to see that $Pr_G(x) = 0$ in case $|x| = 4$ but $x \neq accc$. Then with Theorem 2, we have

$$\begin{aligned} Pr_G(W_1W_2W_1W_2) &= Pr_G(accc)W_1(a)W_2(c)W_1(c)W_2(c) \\ &= 0.008192 \times 0.1 \times 0.3 \times 0.4 \times 0.3 = 0.0000294912. \end{aligned}$$

By direct calculation, we can also obtain that

$$\begin{aligned} Pr_G(W_1W_2W_1W_2) &= Pr(A_1 \Rightarrow W_1A_1A_2A_2)Pr(W_1A_1A_2A_2 \Rightarrow W_1A_2A_1A_2) \\ &\quad \times Pr(W_1A_2A_1A_2 \Rightarrow W_1A_2A_2A_2)Pr(W_1A_2A_2A_2 \Rightarrow W_1W_2A_2A_2) \\ &\quad \times Pr(W_1W_2A_2A_2 \Rightarrow W_1W_2W_1A_2)Pr(W_1W_2W_1A_2 \Rightarrow W_1W_2W_1W_2) \\ &= 0.02 \times 0.8 \times 0.8 \times 0.12 \times 0.16 \times 0.12 = 0.0000294912, \end{aligned}$$

from which we may see the computation tractability that Theorem 2 brings.

5. Concluding remarks

In this paper, we have considered some basic probabilistic models of computation by inputting strings of words, and some new conclusions have been discovered. Specifically, (i) we studied PFAs with input strings of words, and verified that PFAs computing strings of words can be implemented by means of calculating strings of symbols (Theorem 1); (ii) we defined a general type of PTMs and elaborated on PTMs with input strings of words, and particularly demonstrated that PTMs computing strings of words cannot be directly performed through only computing strings of symbols; (iii) we studied PCFGs and thus PRGs with input strings of words, and proved that Theorem 1 does hold for PCFRs and PRGs (Theorem 2, Corollary 1); and we presented a characterization of PRGs in terms of PFAs and the equivalence between PCFGs and their Chomsky and Greibach normal forms, in the sense that the inputs are strings of words. Furthermore, some related problems are raised: (1) does Theorem 1 hold true for two-way PFAs (Remark 2)? (2) can the conventional PTMs by Gill [13] preserve that all computations for a fixed input string of words have the same length (Remark 4)? (3) under what conditions does Theorem 1 hold true for PTMs (Remark 5)? and (4) the question that was addressed in Remark 8.

In [36,34], fuzzy finite automata, fuzzy Turing machines, fuzzy regular grammars, and fuzzy context-free grammars with input strings of words (possibility distributions over input alphabet) were investigated, and it was demonstrated that computation tractability theorems for fuzzy finite automata, fuzzy regular grammars, and fuzzy context-free grammars hold true, but it may not hold for fuzzy Turing machines. In this paper, we dealt with these problems in the sense of probability, and from the results we demonstrated it can be seen that there are somewhat analogies between probabilistic and fuzzy models of computation with input strings of words.

In reality, there are many significant computational models such as probabilistic neural networks [31], residuated lattice-valued automata [26,27], quantum automata [2,3,5,14,19,25], quantum sequential machines [15,24], quantum Turing machines [8,7,14], quantum neural networks [16], and quantum circuits [35], in which the inputs still remain strings of symbols, so, these models with input strings of words are worth considering, especially the computational complexity and the equivalences between models in the sense that the inputs are strings of words, containing the equivalences between probabilistic neural networks and PFAs, as well as between quantum Turing machines and quantum circuits.

Acknowledgements

The authors would like to thank Professor Ying Mingsheng and the referees for careful reading and making helpful comments.

References

[1] L.M. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1023.
[2] A. Ambainis, R. Freivalds, One-way quantum finite automata: strengths, weaknesses and generalizations, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, Palo Alfo, CA, 1998, pp. 332–341.
[3] A. Ambainis, A. Nayak, A. Ta-Shma, U. Vazirani, Dense quantum coding and quantum finite automata, J. ACM 49 (4) (2002) 496–511.
[4] A.V. Aho, J.D. Ullman, The Theory of Parsing, Translation, and Compiling, Prentice-Hall, Englewood Cliffs, NJ, 1973.
[5] A. Broadsky, N. Pippenger, Characterizations of 1-way quantum finite automata, SIAM J. Comput. 31 (2002) 1456–1478.

[6] T.L. Booth, R.A. Thompson, Applying probability measures to abstract languages, IEEE Trans. Comput. C-22 (5) (1973) 442–450.
[7] E. Bernstein, U. Vazirani, Quantum complexity theory, SIAM J. Comput. 26 (5) (1997) 1411–1473.
[8] D. Deutsch, Quantum theory, the Church–Turing principle and the universal quantum computer, Proc. R. Soc. London Ser. A 400 (1985) 97–117.
[9] K. De Leeuw, E.F. Moore, C.E. Shannon, N. Shapiro, Computability by probabilistic machines, in: C.E. Shannon, J. McCarthy (Eds.), Automata Studies, Annals of Mathematics Studies, vol. 34, Princeton University Press, Princeton, NJ, 1956, pp. 183–212.
[10] C.A. Ellis, Probabilistic Languages and Automata, Ph.D. Dissertation, University of IL, Urbana, IL, 1969.
[11] R. Freivalds, Probabilistic two-way machines, in: Proceedings of the International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science, vol. 118, Springer-Verlag, New York, 1981, pp. 33–45.
[12] K.S. Fu, Syntactic Pattern Recognition and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[13] J. Gill, Computational complexity of probabilistic turing machines, SIAM J. Comput. 6 (4) (1977) 675–695.
[14] J. Gruska, Quantum Computing, McGraw-Hill, London, 1999.
[15] S. Gudder, Quantum computers, Int. J. Theoret. Phys. 39 (2000) 2151–2177.
[16] S. Gupta, R.K.P. Zia, Quantum neural networks, J. Comput. System Sci. 63 (2001) 355–383.
[17] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[18] I. Macarie, Closure properties of stochastic languages, Technical Report TR441, Department of Computer Science, University of Rochester, Rochester, NY, 1993.
[19] C. Moore, J.P. Crutchfield, Quantum automata and quantum grammars, Theoret. Comput. Sci. 237 (2000) 275–306.
[20] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[21] H. Ney, Stochastic Grammars and pattern recognition, in: P. Laface, R. De Mori (Eds.), Speech Recognition and Understanding: Recent Advances, Springer, New York, 1992, pp. 319–344.
[22] C.M. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[23] A. Paz, Introduction to Probabilistic Automata, Academic Press, New York, 1971.
[24] D.W. Qiu, Characterization of sequential quantum machines, Internat. J. Theoret. Phys. 41 (5) (2002) 811–822.
[25] D.W. Qiu, Quantum pushdown automata, Int. J. Theoret. Phys. 41 (9) (2002) 1627–1639.
[26] D.W. Qiu, Automata theory based on complete residuated lattice-valued logic(I), Sci. China (F) 44 (6) (2001) 419–429.
[27] D.W. Qiu, Automata theory based on complete residuated lattice-valued logic(II), Sci. China (F) 45 (6) (2002) 442–452.
[28] M.O. Rabin, Probabilistic automata, Inform. Control 6 (1963) 230–245.
[29] A. Salomaa, Probabilistic and weighted grammars, Inform. Control 15 (1969) 529–544.
[30] E.S. Santos, Probabilistic Turing machines and computability, Proc. Amer. Math. Soc. 22 (1969) 704–710.
[31] H.T. Siegelmann, Neural Networks and Analog Computation: Beyond the Turing Limit, Birkhauser, Boston, 1999.
[32] R.A. Thompson, Determination of probabilistic grammars for functionally specified probability-measure languages, IEEE Trans. Comput. C-23 (6) (1974) 603–614.
[33] C.S. Wetherell, Probabilistic languages: a review and some open questions, Comput. Surveys 12 (4) (1980) 361–379.
[34] H. Wang, D.W. Qiu, Computing with words via Turing machines: a formal approach, IEEE Trans. Fuzzy Systems 11 (6) (2003) 707–718.
[35] A.C. Yao, Quantum circuit complexity, in: Proceedings of the 34th IEEE Symposium on Foundations of Computer Science, 1993, pp. 352–361.
[36] M.S. Ying, A formal model of computing with words, IEEE Trans. Fuzzy Systems 10 (5) (2002) 640–652.
[37] L.A. Zadeh, Fuzzy logic, neural networks and soft computing, Commun. ACM 37 (3) (1994) 77–84.
[38] L.A. Zadeh, Fuzzy Logic Computing with Words, IEEE Trans. Fuzzy Systems 4 (2) (1996) 103–111.
[39] L.A. Zadeh, From computing with numbers to computing with words-From manipulation of measurements to manipulation of perceptions, Ann. NY Acad. Sci. 929 (2001) 221–252.